Coppersmith 法の連立方程式への拡張と RSA 暗号への応用について (代数系および計算機科学基礎)

AUTHOR(S):

青野, 良範

CITATION:

青野, 良範. Coppersmith 法の連立方程式への拡張と RSA 暗号への応用について (代数系および計算機科学基礎). 数理解析研究所講究録 2012, 1809: 79-86

ISSUE DATE:

2012-09

URL:

http://hdl.handle.net/2433/194471

数理解析研究所講究録
第1809巻 2012年 79-86

Coppersmith法の連立方程式への拡張とRSA暗号への応用について

青野 良範*

Abstract

本論文では，多変数剰余方程式の小さな解を求めるための手法であるCoppersmith法をとりあげ，その連立方程式への拡張を考える．この手法は，格子アルゴリズムを用いて剰余方程式を代数方程式へと変換するが，格子アルゴリズムへの入力をどのように構成するかによって最終的に求めることのできる解の範囲が決まる．我々は個別の方程式に対応する格子の構成が既に与えられていると仮定し，連立方程式に対応する格子の構成をミンコフスキー和を用いて与える．また，応用としてRSA暗号の安全性解析を行う．

1 Introduction

1996年Coppersmith[4, 5]により提案された，剰余方程式$F(x) \equiv 0 \pmod N$の小さな解を効率的に求めるアルゴリズムは，Howgrave-Graham, Boneh-Durfeeによる多変数への拡張[9]，RSA暗号の安全性解析[1]への応用などから注目され，研究されてきた．このアルゴリズムはサブルーチンとして格子の短いベクトルを求めるアルゴリズム（LLLアルゴリズム[12]など）を利用しているが，剰余方程式が与えられたときにその入力となる格子をどう構成するかという問題は，数多くの応用で問題となってきた．たとえば，JochemszとMay[11]は任意の多変数方程式に対してある程度良い格子の構成を与えているが，いくつかの例に対してもっと良い構成，つまり求めることのできる解の範囲が広いものが知られている．

本論文では，Coppersmith法の連立方程式への拡張を考え，その場合の格子の構成手法を提案する．多くの応用では，まず与えられた連立方程式の両辺を掛けることで1本の方程式に直し，1変数に対するCoppersmith法を適用していた．しかし，これでは掛け算をした段階でいくらか情報が落ちてしまうため，Coppersmith法の真価を発揮することができないと考えられる．我々は多くの応用で個別の方程式に対する良い格子の構成が既に与えられていることに注目し，それらをミンコフスキー和を用いて組み合わせることによって連立方程式に対する格子の構成手法を提案する．

応用として，短い秘密鍵を持つRSA暗号の解析，および部分秘密鍵露出の場合の解析を行う．結果として，ℓ組の短いRSA秘密鍵$d_1, \ldots, d_\ell < N^\beta$に対応する公開鍵$(e_1, N), \ldots, (e_\ell, N)$が与えられたときに，$\beta < (9\ell - 5)/(12\ell + 4)$ならば効率的に暗号が解けてしまうこと，また各$d_k$の下位$\delta n$ビットが既知の場合には$\beta - 0.5\delta + 0.25 < (3\ell - 1)/(3\ell + 1)$ならば暗号が解けてしまうことを証明した．ここで$\beta = 1$と置く，つまり$d$が短いという制限を取ると$\delta/2 > 5/4 - (3\ell - 1)/(3\ell + 1)$となる．ここでたとえば$\ell = 3$とおけば，$\delta > 0.9$．これは，Coppersmith法がうまく動くという仮定の下で，d_kの下位$0.9n$ビットの情報から効率的にNの素因数分解を計算することができること，つまり(e, N)からdの下位ビットを求める問題がNの素因数分解と等価であることを示している．

2 準備

記号 自然数nに対して，集合$\{1, \ldots, n\}$を$[n]$とかく．自然数x，AおよびNに対して，$|x| < A \bmod N \Leftrightarrow [0 \le x < A \text{ or } N - A < x < 0]$と定義する．

整数の組に対して辞書式の順序$\prec$を考える．たとえば，(i_1, i_2)と(i_1', i_2')に対して$(i_1, i_2) \prec (i_1', i_2')$は$i_1 < i_1'$または$[i_1 = i_1' \text{ and } i_2 < i_2']$と定義される．この記号を単項式同士の比較にも用い，$x_1^{i_1} x_2^{i_2} \prec x_1^{i_1'} x_2^{i_2'} \Leftrightarrow (i_1, i_2) \prec (i_1', i_2')$とする．以上の記号を一般の$n$-項と$n$変数単項式に拡張して用いる．変数をあらわす記号は特に断りがない限

*独立行政法人 情報通信研究機構，〒184-8795 東京都小金井市貫井北町4-2-1 ネットワークセキュリティ研究所 セキュリティ基盤研究室

り $x_1, x_2, \cdots$ と y を使い，これらの順位を $x_1 > x_2 > \cdots > y$ と定めて単項式同士の比較を行う．たとえば、x_1, x_2, x_3, y の単項式 $x_1^2x_2^3y$ と $x_2^2x_3$ を比較するときには，$(2,3,0,1) \succ (0,2,1,0)$ なので $x_1^2x_2^3y \succ x_2^2x_3$ となる．

以上のように定められた単項式の順序に対して，多項式 $f(x_1, \ldots, x_\ell, y)$ に現れる係数がゼロでない項で最大のものを $ax_1^{i_1} \cdots x_\ell^{i_\ell} y^j$ と書いたとき，これを頭項 (head term) とよび，$HT(f)$ とかく．また，a，$y^j x_1^{i_1} \cdots x_\ell^{i_\ell}$ および $(i_1, \ldots, i_n, j)$ をそれぞれ頭係数 (head coefficient)，頭単項式 (head monomial)，頭指数と呼び $HC(f)$，$HM(f)$，$HI(f)$ であわらす[1]

ミンコフスキー和　A と B を $\mathbb{Z}^\ell$ の有限部分集合とする．このとき，これらの集合のミンコフスキー和は $A+B = \{(a_1+b_1, \ldots, a_\ell+b_\ell) : (a_1, \ldots, a_\ell) \in A, (b_1, \ldots, b_\ell) \in B\}$ で定義される．

2.1 Coppersmith 法の概要

本論文では，Coppersmith 法 [4] と呼ばれる，剰余方程式の小さな整数解を求める手法を扱う．この手法の目的は，整数係数多変数多項式 $F(x,y)$，自然数 W，範囲指定パラメータ X, Y が与えられたときに

$$F(x,y) \equiv 0 \pmod{W} \tag{1}$$

をみたす整数 (x,y) で，$|x| < X$ かつ $|y| < Y$ をみたすものを効率的に全て求めることである．以下，解きたい方程式を (1)，範囲指定パラメータを X, Y と固定して説明するが，3 変数以上の場合にも同様に問題が定義できる．

Coppersmith 法の基本的な戦略を説明する．まず，自然数 m を固定し，

$$\forall x, y \left[\begin{array}{l} F(x,y) \equiv 0 \pmod{W} \\ \Rightarrow g(x,y) \equiv 0 \pmod{W^m} \end{array} \right] \tag{2}$$

をみたす多項式 $g(x,y)$ の集合 L を考える．条件の形より，この集合は多項式の格子をなす，つまり $\forall g_1, g_2 \in L \Rightarrow g_1 + g_2 \in L$ がなりたつ．次に，この多項式の格子の中から

$$\forall x, y, |x| < X, |y| < Y \left[\begin{array}{l} h(x,y) \equiv 0 \pmod{W^m} \\ \Rightarrow h(x,y) = 0 \end{array} \right] \tag{3}$$

をみたす $h(x,y)$ で代数的に独立なものを変数の数（ここでは 2 つ）求めることでもとの問題を整数上の連立代数方程式の形に直す．最後に，連立方程式を終結式やグレブナー基底の手法を用いて解く．

[1]頭項，頭係数，頭単項式の用語は文献 [14] に依った．

後に述べるように，係数の小さい多項式は (3) をみたすので，格子 L の中の「短い」元を求めることで目的の多項式が見つかる．格子の短い元をみつける効率的なアルゴリズムには，例えば LLL アルゴリズム [12] 等があるが，これらのアルゴリズムは主にユークリッド空間 $\mathbb{R}^n$ 内の格子の小さい ℓ^2 ノルムを持つベクトルを求めるように設計されているため，多項式の格子には直接適用できない．そのため，以下のように多項式とベクトルの変換を考える．

多項式とベクトルの変換　多項式 $g(x,y) = \sum_{i,j} a_{i,j} x^i y^j$ に対して，パラメータ X, Y を用いたベクトル化を以下のように定義する．

$\mathcal{V}(g; X, Y) = (a_{0,0}, a_{0,1}X, \ldots, a_{i_w, j_w} X^{i_w} Y^{i_w})$

つまり，$g(x,y)$ の各項 $a_{i,j}x^iy^j$ をベクトルの各成分 $a_{i,j}X^iY^j$ に対応させた写像であり，g に関して線形写像になる．ここで，添え字の列 $\{(i_k, j_k)\}_{k=1}^w$ は $g(x,y)$ の非ゼロ項を全て尽くすように取る．後に複数の多項式をベクトルに変換する必要が出てくるが，その場合には適切な添え字の列を固定して変換を行うものとする．また，このベクトルのユークリッドノルム $|\mathcal{V}(g; X, Y)|$ を，パラメータ X, Y に関する多項式のノルムとして定義し，$||g||_{XY}$ とかく．

このとき，以下の補題がなりたつ．

補題 1. *([9],[11])* X, Y, W を自然数，$h(x,y)$ を整数係数多項式で，w 個の単項式の和でかけ，かつ $||h(x,y)||_{XY} < W/\sqrt{w}$ をみたすものとする．このとき，

$\forall x, y, |x| < X, |y| < Y$
$[h(x,y) \equiv 0 \pmod{W} \Leftrightarrow h(x,y) = 0]$.

つまり，格子 L のなかで，上の補題をみたす独立な多項式を変数の数だけ求めることができれば，目的は達成される．そのために，L の元をベクトル化したユークリッド空間内の格子を考え，格子アルゴリズムを用いてユークリッド空間内の短い格子ベクトルを求める．

格子　ここでは，ユークリッド空間の格子と多項式のなす格子を定義する．$\mathbb{R}^c$ の中の一次独立なベクトル $\mathbf{b}_1, \ldots, \mathbf{b}_c$ に対して，これらを基底とする格子を以下の集合で定義する．

$L(\mathbf{b}_1, \ldots, \mathbf{b}_c)$
$= \{a_1\mathbf{b}_1 + \cdots + a_c\mathbf{b}_c \ : \ a_k \in \mathbb{Z} \text{ for } k \in [c]\}$

つまり，基底ベクトルの整数係数一次結合の集合である．このとき，基底ベクトルの本数 c を格子の次元と呼ぶ．

Coppersmith 法の多くの応用では，格子中の短いベクトルを求めるために LLL アルゴリズム [12] をサブルーチンとして用いている．このアルゴリズムは，与えられた格子基底に対して LLL-縮小基底と呼ばれる良い基底を効率的に計算する．以下の定理で与えられるように LLL-縮小基底の最初の方のベクトルは短いという性質がある．

定理 1. *[3]* L を c 次元の格子，$\mathbf{v}_1,\ldots,\mathbf{v}_c$ をその *LLL* 縮小基底とする．このとき，以下の不等式がなりたつ．

$$||\mathbf{v}_i|| \le 2^{\frac{(c(c-1)+(i-1)(i-2)}{4(c-i+1)}} |\det(L)|^{1/(c-i+1)} \quad (4)$$

ただし，$\det(L)$ は格子の行列式であり，格子基底 $\mathbf{b}_1,\ldots,\mathbf{b}_c$ に対応する *Gram-Schmidt* 直交基底 $\mathbf{b}_1^*,\ldots,\mathbf{b}_c^*$ を用いて，$\det(L)=\prod_{i=1}^{c}||\mathbf{b}_i^*||$ と定義される．

$\mathbb{R}^c$ 内の格子と同様に，$\mathbb{Z}$ 上独立な整数係数多項式の集合 $\mathbf{G}=\{g_1,\ldots,g_c\}$ を基底とした多項式の格子を以下で定義する．

$$L(\mathbf{G}) = L(g_1,\ldots,g_c)$$
$$= \{a_1g_1+\cdots+a_cg_c \ : \ a_i\in\mathbb{Z} \text{ for } k\in[c]\}$$

また，この格子の基底多項式をパラメータ X,Y でベクトル化したものを基底とした格子 $L(\mathcal{V}(g_1;X,Y),\ldots,\mathcal{V}(g_c;X,Y))$ を $L(\mathbf{G};X,Y)$ とかき，その行列式を $\det(\mathbf{G};X,Y)$ とかく．

Coppersmith 法の動作条件　問題の方程式と範囲パラメータ X,Y を固定したときに，もしも (2) をみたす c 次元，項数 w 以下の多項式格子 $L(\mathbf{G})$ で

$$2^{c/4}\det(\mathbf{G};X,Y)^{1/c} < N^m/w \quad (5)$$

がなりたつようなものが構成できたとする．このとき，定理 1 の $i=1,2$ の場合から，$L(\mathbf{G};X,Y)$ の LLL-縮小基底の最初の 2 本 $\mathbf{v}_1$ と $\mathbf{v}_2$ の長さは N^m/w よりも小さくなるため，対応する多項式は補題 1 の仮定をみたす．そして，$\mathbf{v}_1$ と $\mathbf{v}_2$ を逆変換する，つまり $i=1,2$ に対して $\mathbf{v}_i=\mathcal{V}(h_i;X,Y)$ をみたすような多項式を求め，これらを連立方程式 $h_1=h_2=0$ として解くことで効率的に小さな整数解を求めることができる．

以上が Coppersmith 法の概要であり，アルゴリズムとしてまとめると図 1 となる．

多くの応用では，方程式 $F(x,y)\equiv 0 \pmod W$ が与えられたときに，できる限り大きな X,Y に対して (5) をみたす格子 $\mathbf{G}$ を構成することが問題となる．実際に解析を行うときには，いくつかの小さい因子を無視した以下の条件を動作条件とする．

$$\det(\mathbf{G};X,Y)^{1/c} < N^m \quad (6)$$

最後の連立方程式 $h_1=h_2=0$ が解けるためには，多項式が代数的に独立でなければならないが，これは理論的に保証できない．多くの多変数 Coppersmith 法の論文と同様に，この部分の独立性は保証されると仮定し，計算機実験によりその仮定を正当化する．

Input: $F(x,y)\in\mathbb{Z}[x,y]$, W, X,Y; パラメータ $c\ge 1$, $m\ge 2$;
Output: $|x|<X$ かつ $|y|<Y$ をみたす $F(x,y)\equiv 0 \pmod W$ の全ての整数解
Step 1: $F(x,y)$ と W に関して，(2) をみたす $\mathbb{Z}$ 上独立な多項式 $g_1(x,y),\ldots,g_k(x,y)$ を選び，それを基底とした多項式の格子 $L(\mathbf{G})$ を構成
Step 2: 多項式の格子をユークリッド空間の格子 $L(\mathbf{G};X,Y)$ に変換し，その LLL-縮小基底 $\mathbf{v}_1,\ldots,\mathbf{v}_k$ を求める
Step 3: $h_1(x,y),h_2(x,y)$ をそれぞれ $\mathbf{v}_1,\mathbf{v}_2$ に対応する多項式とする．連立代数方程式 $h_1(x,y)=h_2(x,y)=0$ を解き，対応する小さな整数解を全て出力する．

図 1: Coppersmith 法のアルゴリズム

2.2 RSA 方程式

以下，Boneh-Durfee[1] が短い秘密鍵を持つ RSA 暗号の解析のために導入した方程式とその適用限界を説明する．RSA 暗号の記号は標準的なものを用いる．つまり，大きな素数 p と q の積を N と置き，公開鍵指数 e と秘密鍵指数 d の間には関係式 $ed\equiv 1 \pmod{\varphi(N)}$ がある．また，[1] と同様に $e\approx N$, $p\approx q$ がなりたつものとする．特に，$p+q<3N^{0.5}$ を仮定する．d は N よりも非常に小さいと仮定し，実数 $\beta\in(0,1)$ を $N^\beta=d$ をみたすものとする．

e と d の関係式から，適当な整数 k,s に対して，$ed+k(N+s)=1$ がなりたつことがわかるので，$(x,y)=(k,s)$ は以下の RSA 方程式の一組の解となる．

$$f_{\mathrm{RSA}}(x,y) = -1+x(y+N)\equiv 0 \pmod e \quad (7)$$

逆に，$\beta<0.5$ であれば $|x|<N^\beta$ と $|y|<3N^{0.5}$ をみたす解はほとんどの場合ただ一つに定まり，上

記 (k,s) となることがわかる．もしもこの範囲にある解を求めることができれば，$s=-(p+q)$ なので N の素因数分解ができ，そこから RSA 暗号を解くことができる．

3 Coppersmith 法の連立方程式への拡張

この節では，Coppersmith 法の連立方程式への拡張を提案する．簡単のため，方程式の本数は2本とし，変数 y のみが共通であるような以下の連立方程式を考えるが，多変数への一般化は容易である．

$$\begin{aligned} F_1(x_1,y) &\equiv 0 \pmod{W_1} \\ F_2(x_2,y) &\equiv 0 \pmod{W_2} \end{aligned} \tag{8}$$

このとき，$|x_1|<X_1$，$|x_2|<X_2$ および $|y|<Y$ をみたす整数解を全て求めることを目標とする．

アルゴリズム自体は図1に挙げたものとほぼ同様だが，入力が連立方程式になっていることと (2) に対する条件だけが異なる．つまり，アルゴリズムの入力は2つの多項式 $F_1(x_1,y)$ と $F_2(x_2,y)$，対応する法 W_1 と W_2，解の範囲を示す X_1,X_2,Y，そしてパラメータ c と m であり，Step 1 における3変数多項式 $g_i(x_1,x_2,y)$ に対する条件は

$$\forall x_1,x_2,y\in\mathbb{Z},\quad \left[\begin{array}{l} F_1(x_1,y)\equiv 0 \pmod{W_1} \\ F_2(x_2,y)\equiv 0 \pmod{W_2} \\ \Rightarrow g_i(x_1,x_2,y)\equiv 0 \pmod{(W_1W_2)^m} \end{array}\right] \tag{9}$$

となる．集合 $\mathbf{G}=\{g_1,\ldots,g_c\}$ 基底とする多項式の格子 $L(\mathbf{G})$ を考え，Step 2 で $L(\mathbf{G};X_1,X_2,Y)$ の LLL-縮小基底を計算する．

第2節における解析と同様の仮定を置くことで，アルゴリズムが動作するための条件が $\det(\mathbf{G};X_1,X_2,Y)^{1/c}<(W_1W_2)^m$ であることが導出できる．次の節で多項式の格子をどのように構成するかを説明する．

3.1 ミンコフスキー和を用いた格子の構成

Coppersmith 法の応用において，連立方程式を解く場合には既に個別の方程式に対する良い多項式の格子が得られていることが多い．そのため，我々もこれを仮定し，既知の格子を組み合わせて連立方程式用の格子を構成する手法を与える．簡単のため，引き続き前節で用いた方程式を使うが，一般化は容易である．

いま，$k=1,2$ に対して $F_k(x_k,y)\equiv 0 \pmod{W_k}$ を解くための多項式の格子を $L(\mathbf{G}_k)$，その基底を $\mathbf{G}_k=\{g_1^{(k)},\ldots,g_{c_k}^{(k)}\}$ として，これらの組み合わせにより連立方程式を解くための格子を構成する．

全ての $\ell_1\in[c_1],\ell_2\in[c_2]$ に対して多項式 $g_{\ell_1}^{(1)}\cdot g_{\ell_2}^{(2)}$ を考えると，これは条件 (9) をみたすので，集合

$$\mathcal{A}=\left\{\sum_{\ell_1,\ell_2} a_{\ell_1,\ell_2}g_{\ell_1}^{(1)}g_{\ell_2}^{(2)} : a_{\ell_1,\ell_2}\in\mathbb{Z}\right\}$$

を定義すると，これは連立方程式を解くための多項式の格子となる．しかし，一般に $\{g_{\ell_1}^{(1)}g_{\ell_2}^{(2)}\}_{\ell_1,\ell_2}$ は $\mathbb{Z}$ 上一次独立ではないので具体的な基底の形がわからず，行列式の解析ができない．そこで，我々は $\mathcal{A}$ の部分格子の基底をミンコフスキー和を用いて定義する．

いま，基底 $\mathbf{G}_k$ は strictly increasing degree order，つまり $HT(g_1^{(k)})<\cdots<HT(g_{c_k}^{(k)})$ をみたしていると仮定する．そうでない場合でも，ガウスの消去法によりこの条件をみたす等価な基底が計算可能である．まず，$k=1,2$ に対して，基底の頭指数の集合 $I_k=\{HI(g_\ell^{(k)}):\ell\in[c_k]\}$ を考え，そのミンコフスキー和を $I_+=I_1+I_2$ とする．I_+ の各元 (i_1,i_2,j) に対して，(9) をみたす多項式を以下のように定義する．

$$g_{i_1,i_2,j}^{+}=\sum_{(*)} a_\lambda g_\lambda^{(1)}g_{\lambda'}^{(2)} \tag{10}$$

ただし，和 $(*)$ は $HM(g_\lambda^{(1)}g_{\lambda'}^{(2)})=x_1^{i_1}x_2^{i_2}y^j$ となる全ての組み合わせに対して取り，係数 a_λ は $LC(g_{(i_1,i_2,j)}^{+})=GCD_{(*)}(LC(g_\lambda^{(1)}g_{\lambda'}^{(2)}))$ となるように取る．つまり，考えうる組み合わせの中で，最高次の係数の絶対値が非ゼロかつ最小になるように取る．

このように定義した多項式の集合 $G_+=\{g_{i_1,i_2,j}^{+}:(i_1,i_2,j)\in I_+\}$ を基底とする格子 $L(G_+)$ を $L(G_1)$ と $L(G_2)$ のミンコフスキー和格子と呼ぶ．明らかに，$L(G_+)\subset\mathcal{A}$ である．

3.2 下三角行列で表現される格子のミンコフスキー和

この節では，$k=1,2$ に対してあるインデックスの列 $\{(i_1(k,\ell),i_2(k,\ell),j(k,\ell))\}_{\ell=1}^{c_k}$ が存在し，多項式格子の基底が

$$g_\ell^{(k)}=\sum_{\ell'=1}^{\ell} a_{\ell,\ell'}x_1^{i_1(k,\ell')}x_2^{i_2(k,\ell')}y^{j(k,\ell')},\quad a_{\ell,\ell}\neq 0$$

とかける．つまり $I_k=\{(i_1(k,\ell),i_2(k,\ell),j(k,\ell)):\ell\in[c_k]\}$ となって上記インデックス列と一致する場合を考える．このとき，格子 $L(\mathbf{G}_k;X_1,X_2,Y)$

は下三角行列になる．このような形の格子は，行列式の解析が容易なため多くの応用で使われる．これらの格子のミンコフスキー和も下三角行列になることを示す．

定理 2. $i = 1,2$ に対して $\mathbf{G}_i = \{g_1^{(i)}, \ldots, g_{k_i}^{(i)}\}$ が *strictly increasing degree order*、かつ $L(\mathbf{G}_i; X_1, X_2, Y)$ が下三角行列とする．このとき，格子のミンコフスキー和 $\mathbf{G}_+ = \{g^+_{i_1,i_2,j} : (i_1,i_2,j) \in I_+\}$ をインデックスの順序で並べた基底から $L(G_+; X_1, X_2, Y)$ を構成すると，下三角行列になる．

証明. 任意の $(i_1,i_2,j) \in I_+$ に対して，

$g^+_{i_1,i_2,j} = a x_1^{i_1} x_2^{i_2} y^j$
$+(x_1^{i_1} x_2^{i_2} y^j$ よりも低く，指数が I_+ に入っている単項式の線形結合)

であることを示す．作り方より，(10) の (*) をみたす $g_\lambda^{(1)} g_{\lambda'}^{(2)}$ が上の形でかけることをいえばよい．$g_\lambda^{(1)} g_{\lambda'}^{(2)}$ を単項式の形に展開した時に出てくる項は，$g_\lambda^{(1)}$ のある非ゼロ項 $A x_1^{\bar{i}_1} y^{\bar{j}}$ と $g_{\lambda'}^{(2)}$ のある非ゼロ項 $B x_2^{\bar{i}_2} y^{\bar{j}'}$ の積の和としてかける．仮定より，$(\bar{i}_1, 0, \bar{j}) \in I_1$ かつ $(0, \bar{i}_2, \bar{j}') \in I_2$ なので $(\bar{i}_1, \bar{i}_2, \bar{j} + \bar{j}') \in I_+$．また，$(\bar{i}_1, 0, \bar{j}) \preceq HI(g_\lambda^{(1)})$ かつ $(0, \bar{i}_2, \bar{j}') \preceq HI(g_{\lambda'}^{(2)})$ なので $(\bar{i}_1, \bar{i}_2, \bar{j} + \bar{j}') \preceq HI(g_\lambda^{(1)}) + HI(g_{\lambda'}^{(2)}) = HI(g^+_{i_1,i_2,j})$．また，最高次の係数は作り方から非ゼロであるので，$L(G_+; X_1, X_2, Y)$ は下三角行列になる．□

4 2つ以上の短い秘密鍵に対するRSA暗号の解析

以上の議論を使って，RSA 暗号の解析を行う．Howgrave-Graham と Seifert[10]，および Sarkar と Maitra[16, 17] の論文に従い，以下を仮定する．攻撃者は $\ell \geq 2$ 組の RSA 公開鍵 $(e_1, N), \ldots, (e_\ell, N)$ を持ち，対応する秘密鍵 d_k は全て N^β よりも小さいとする．このとき，$\beta < (\ell - 0.5)/\ell$ ならば連立方程式

$$\begin{aligned} F_1(x_1, y) &= -1 + x_1(y+N) \equiv 0 \pmod{e_1} \\ &\vdots \\ F_\ell(x_\ell, y) &= -1 + x_\ell(y+N) \equiv 0 \pmod{e_\ell} \end{aligned} \tag{11}$$

の $|x_k| < X_k := N^\beta$ かつ $|y| < Y := 3N^{0.5}$ をみたす解 $(x_1, \ldots, x_\ell, y)$ はほとんどの場合一意に定まり，そこから N の素因数分解ができる．

また，前節と同様に各 $F_k(x_k, y) \equiv 0 \pmod{e_k}$ に対応する多項式の格子はあらかじめ与えられているものとする．ここでは，[1] のもっとも単純なものを採用する．$g_{i,j}^{(k)} = x_k^{i-j} F_k(x_k, y) e_k^{m-j}$ とおくと，この多項式は $F_k(x_k, y) \equiv 0 \pmod{e_k}$ に関して (2) をみたす．また，$HM(g_{i,j}^{(k)}) = x_k^i y^j$ となるので，これらの多項式は $\mathbb{Z}$ 上独立．自然数 m を固定して集合 $\mathbf{G}_k = \{g_{i,j}^{(k)} : (i,j) \in \mathbb{Z}^2, 0 \leq j \leq i \leq m\}$ を考えると，これは格子基底になり，しかもそのベクトル化 $L(\mathbf{G}_k; X_k, Y)$ は下三角行列になる．

以下，格子 $\mathbf{G}_1, \ldots, \mathbf{G}_\ell$ のミンコフスキー和 $\mathbf{G}_+$ の形を具体的に求める．登場する変数は $x_1, \ldots, x_\ell, y$ なので，$HI(g_{i,j}^{(k)}) = (0, \ldots, 0, i, 0, \ldots, 0, j)$ (k 番目の成分が i，$\ell+1$ 番目の成分が j，それ以外はゼロ．) よって，$\mathbf{G}_k$ に対応するインデックスの集合は $I_k = \{(0, \ldots, 0, i, \ldots, 0, j) : (i,j) \in \mathbb{Z},\ 0 \leq j \leq i \leq m\}$ となり，そのミンコフスキー和は

$$\begin{aligned} I_+ &= I_1 + \cdots + I_\ell \\ &= \{(i_1, \ldots, i_\ell, j) : 0 \leq i_1, \ldots, i_\ell \leq m \\ &\quad \text{and } 0 \leq j \leq i_1 + \cdots + i_\ell\} \end{aligned}$$

となる．各 $(i_1, \ldots, i_\ell, j) \in I_+$ に対して，(9) をみたす多項式を

$$g_{i_1,\ldots,i_\ell,j} = \sum_{j_1,\ldots,j_\ell} a_{j_1,\ldots,j_\ell} \cdot g_{i_1,j_1}^{(1)} g_{i_2,j_2}^{(2)} \cdots g_{i_\ell,j_\ell}^{(\ell)}$$

で定義する．ただし，このときの和は (10) と同様に $HM(g_{i_1,j_1}^{(1)} g_{i_2,j_2}^{(2)} \cdots g_{i_\ell,j_\ell}^{(\ell)}) = x_1^{i_1} \cdots x_\ell^{i_\ell} y^j$ となる組み合わせ，つまりここでは $0 \leq j_k \leq i_k$ かつ $j_1 + \cdots + j_\ell = j$ をみたす全ての $(j_1, \ldots, j_\ell)$ に対して和を取る．また，係数 $a_{j_1,\ldots,j_\ell}$ は

$$\begin{aligned} &HC(g_{i_1,\ldots,i_\ell,j}) \\ &= GCD_{j_1,\ldots,j_\ell} \left(HC(g_{i_1,j_1}^{(1)} g_{i_2,j_2}^{(2)} \cdots g_{i_\ell,j_\ell}^{(\ell)}) \right) \end{aligned}$$

となるようにとる．いま $HC(g_{i_1,j_1}^{(1)} g_{i_2,j_2}^{(2)} \cdots g_{i_\ell,j_\ell}^{(\ell)}) = e_1^{m-j_1} \cdots e_\ell^{m-j_\ell}$ であり，j_k は 0 から $\min(i_k, j)$ まで動くので，上記最大公約数は $e_1^{m-\min(i_1,j)} \cdots e_\ell^{m-\min(i_\ell,j)}$ となる．右辺がこの値になるように係数 $a_{j_1,\ldots,j_\ell}$ を取る．

定理 2 より，$L(\mathbf{G}_+; X_1, \ldots, X_\ell, Y)$ は下三角行列になり，インデックス $(i_1, \ldots, i_\ell, j)$ に対応する対角成分は $\mathcal{V}(HT(g_{i_1,\ldots,i_\ell,j}); X_1, \ldots, X_\ell, Y) = e_1^{m-\min(i_1,j)} \cdots e_\ell^{m-\min(i_\ell,j)} X_1^{i_1} \cdots X_\ell^{i_\ell} Y^j$ となる．よって，行列式は

$$L(\mathbf{G}_+; X_1, \ldots, X_\ell, Y) = \prod_{(i_1,\ldots,i_\ell,j) \in I_+} \left[\prod_{k=1}^{\ell} (e_k^{m-\min(i_k,j)} X_k) \times Y^j \right]$$

と計算できる．これを Coppersmith 法が動作するための条件 $\det(\mathbf{G}_+;X_1,\ldots,X_\ell,Y)^{1/|I_+|} < (e_1\cdots e_\ell)^m$ に代入して，近似式 $e_1\approx e_2\approx\cdots\approx e_\ell\approx N$，$X_1=\cdots=X_\ell=N^\beta$ および $Y\approx N^{0.5}$ をつかうと，

$$\sum_{(i_1,\ldots,i_\ell,j)\in I}\left[0.5j+(i_1+\cdots+i_\ell)\beta -\sum_{k=1}^{\ell}\min(i_k,j)\right]<0 \tag{12}$$

となり，左辺を計算すると

$$\left(-\frac{3}{16}\ell^2+\frac{5}{48}\ell+\left(\frac{\ell^2}{4}+\frac{\ell}{12}\right)\beta\right)m^{\ell+2} +o(m^{\ell+2})<0$$

となる．よって，m が十分大きいときにこの条件は

$$\beta<\frac{9\ell-5}{12\ell+4}$$

となる．

5 部分秘密鍵露出攻撃への応用

次に，RSA 暗号の部分秘密鍵露出攻撃を取り上げる．これは，RSA 秘密鍵が短く，さらに部分情報が露出している場合の安全性解析を行うものである．

本節で取り上げる状況は，ℓ 組の RSA 公開鍵 (e_k,N) と秘密鍵 d_k に対して，d_k が小さく $(<N^\beta)$，かつ d_k の下位 δn ビットが露出している状況を考える．現実にはこのような状況がおこることは考えにくいが，$\beta=1$ の場合，(e,N) から d の下位ビットを計算するアルゴリズムが存在すればその出力を用いて多項式時間で N の素因数分解ができる，という議論をするためにこの仮定を置いている．

いま，$M=2^{\lfloor\delta n\rfloor}$，露出部分を $\widetilde{d_k}$ とおくと，前節までの議論と同様に，連立方程式

$$\begin{aligned} F_1(x_1,y)&=e_1\widetilde{d_1}-1+x_1(y+N)\equiv 0 \pmod{e_1M}\\ &\vdots\\ F_\ell(x_\ell,y)&=e_\ell\widetilde{d_\ell}-1+x_\ell(y+N)\equiv 0 \pmod{e_\ell M}\end{aligned} \tag{13}$$

の $|x_1|,\ldots,|x_\ell|<N^\beta$ かつ $|y|<3N^{0.5}$ をみたす解 $(x_1,\ldots,x_\ell,y)$ は $\beta-\delta<(\ell-0.5)/\ell$ のとき，ほとんどの場合唯一に定まり，N の素因数分解ができることが示せる．

前節と同様に $g_{i,j}^{(k)}=x_k^{i-j}(F_k(x_k,y))^j\,(e_kM)^{m-j}$ とおくと，(2) をみたし，$\mathbf{G}_k=\{g_{i,j}^{(k)}:(i,j)\in\mathbb{Z}^2, 0\le j\le i\le m\}$ は格子基底になり，$L(\mathbf{G}_k;X_k,Y)$ も下三角行列になる．いま，連立方程式 (13) の形は前節の (11) と比べると定数項と法のみが異なるので，$I_1,\ldots,I_\ell,I_+$ は前節と同じものとなる．各 $(i_1,\ldots,i_\ell)\in I_+$ に対して $g_{i_1,\ldots,i_\ell,j}$ を構成すると

$$\begin{aligned}&HT(g_{i_1,\ldots,i_\ell,j})\\ &=e_1^{m-\min(i_1,j)}\cdots e_\ell^{m-\min(i_\ell,j)}M^{\ell m-j}x_1^{i_1}\cdots x_\ell^{i_\ell}y^j\end{aligned}$$

となり，

$$\begin{aligned}&L(\mathbf{G}_+;X_1,\ldots,X_\ell,Y)\\ &=\prod_{(i_1,\ldots,i_\ell,j)\in I_+}\left[e_1^{m-\min(i_1,j)}\cdots e_\ell^{m-\min(i_\ell,j)}\right.\\ &\left.\times M^{\ell m-j}X_1^{i_1}\cdots X_\ell^{i_\ell}Y^j\right]\end{aligned}$$

と計算できる．よって，$e_k\approx N, X_k\approx N^\beta, Y\approx N^{0.5}$ の近似を使うと，Coppersmith 法がうまく動くための条件は，

$$\sum_{(i_1,\ldots,i_\ell,j)\in I}\left[(0.5-\delta)j+(i_1+\cdots+i_\ell)\beta -\sum_{k=1}^{\ell}\min(i_k,j)\right]<0 \tag{14}$$

となり，m が十分大きなときには

$$\beta-\frac{\delta}{2}+\frac{1}{4}<\frac{3\ell-1}{3\ell+1}$$

となる．特に，$\beta=1$ の場合には秘密鍵の長さが短いという制限が消え，$\delta/2>5/4-(3\ell-1)/(3\ell+1)$ となる．$\ell\ge 3$ のときにこの式は意味を持ち，例えば $\ell=3$ の場合には $\delta>0.9$ になる．つまり，(e_k,N) および対応する d_k の下位 $0.9n$ ビットが 3 組与えられれば，それらの情報から Coppersmith 法を用いて多項式時間で N の素因数分解が可能であることを示している．つまり，(e_k,N) から d_k の下位 $0.9n$ ビットを計算する問題の難しさは，Coppersmith 法が動作するという仮定の下で N の素因数分解と等価であるということが言える．ℓ を十分大きく取れば $\delta>0.5+\varepsilon$ であるので，上の定数 0.9 は 0.5 にいくらでも近づけることができる．

6 計算機実験

提案した RSA 暗号の安全性解析アルゴリズムの有効性を検証するため，計算機実験を行った．使用した計算機は標準的なワークステーションで，16GB の RAM と Intel Xeon 5675 プロセッサ (3.07GHz) を 2 個搭載している．提案アルゴリズムの実装には C++言語を用いた．LLL 縮小基底の計算には Shoup の NTL ライブラリバージョン 5.5.2[13] を GMP ライブラリ 5.0.4[7] でコンパイルしたものを用いた．多項式の計算には GiNaC ライブラリバー

ジョン 1.6.2[8] を使用した．コンパイルは g++コンパイラ 4.5.4 を使用し，-O3 オプションを用いた．また，最終的な終結式の計算には Maple 15 を用いた．実験は全て Windows 7 上で動作し，シングルスレッドのプログラムを用いた．

パラメータ: ℓ: RSA 鍵の個数，n: RSA 暗号のビット長，δ: 秘密鍵と法のビット長比率

Step 1: (RSA インスタンスの作成) $\lfloor n/2 \rfloor$ ビットの疑素数 p と q を生成し，$N = pq$ とおく．$\gcd(d_i, (p-1)(q-1)) = 1$ をみたす ℓ 個の $\lfloor \delta n \rfloor$ ビット奇数 $d_1, \ldots, d_\ell$ をランダムに生成し，秘密鍵とする．対応する公開鍵 $e_i = d_i^{-1} \pmod{(p-1)(q-1)}$ を計算．RSA 方程式 $f_k(x_k, y) = -1 + x_k(N+y) \pmod{e_k}$ を設定し，秘密鍵に対応する小さな解 $\bar{x_k} = (1 - e_k d_k)/((p-1)(q-1))$ と $\bar{y} = 1 - p - q$ を設定．

Step 2: 解の上界 $X_k = \lfloor N^\delta \rfloor$ と $Y = \lfloor 3N^{0.5} \rfloor$ を設定．4 節のやり方で，多項式の格子 $L(\mathbf{G})$ を構成し，そのベクトル化 $L(\mathbf{G}_+; X_1, \ldots, X_\ell, Y)$ に対して LLL アルゴリズムを適用．

Step 3: 縮小後の基底の最初の $\ell + 1$ 本のベクトル $\mathbf{v}_1, \ldots, \mathbf{v}_{\ell+1}$ から，対応する多項式 $h_1, \ldots, h_{\ell+1}$ を計算する．

Step 4: まず，(i) Step 1 で作った解を代入し，$h_i(\bar{x_1}, \ldots, \bar{x_\ell}, \bar{y}) = 0$ を $i \in [\ell + 1]$ に対して確認する．次に，(ii)$h_1, \ldots, h_{\ell+1}$ の終結式を計算し，これらの多項式の独立性を確認する．これら 2 つのチェックを通過したら，実験は成功したとみなす．

図 2: 計算機実験の手順

計算機実験の手順を図 2 に示す．ほぼ図 1 と同じだが，いくつか説明を加える．Step 1 の疑素数の生成は，ランダムに奇数を発生させた上で Euler-Jacobi 素数判定を $a = 2, 3, 5, 7$ に対して行い，通過したものを採用している．Step 2 の LLL アルゴリズムは，NTL ライブラリの `LLL_XD(L,0.99,0,0,1)` を用いた．Step 4 における解の検証は最初にすべての $i \in [\ell + 1]$ に対して $h_i(\bar{x_1}, \ldots, \bar{x_\ell}, \bar{y}) = 0$ を確認し，ひとつでも値がゼロにならないものがあれば実験失敗としている．次に，これらの多項式の終結式を素数 P を法として計算することで変数を消去し，最終的に計算された 1 変数多項式 $R(y)$ に解 $\bar{y}$ を代入して mod P でゼロになるかどうかを，異なる 3 つの素数に対して確認した．

ℓ	m	β_{thm}	dim.	β_{exp}	LLL-time
2	2	0.386	27	0.386	10.5 sec
2	3	0.405	64	0.406	30 min. 44 sec.
2	4	0.416	125	0.414	20 hr. 26 min.
3	2	0.464	108	0.464	3 hr. 17 min.

表 1: 小さい l, m に対する具体的な β の上界，および 1024 ビット RSA に対する実験結果

6.1 実験パラメータと結果

小さい秘密鍵に対する実験は以下のように行った．具体的な m, ℓ に対して式 (12) を計算すると，β の一次式になり，これをみたす最大の β が，対応する $L(\mathbf{G})$ を使って解くことのできる RSA 秘密鍵の範囲の理論値をあらわす．ただし，これは N を十分大きくとった場合の値であり，前節までの議論から細かい因子を無視したものであるので正確な値ではない．

実験では N の長さを 1024 ビット，$\ell = 2$ に対して $m = 2, 3, 4$，$\ell = 3$ に対して $m = 2$ として，理論値周辺の β を 0.002 刻みで動かし，各 β に対して，1 回ずつ実験を行った．表 1 に各パラメータに対する β の理論値，解くことのできた最大の β，格子の次元，LLL にかかった時間を示す．

次に，部分秘密鍵露出の場合の実験を行った．手順は図 2 で挙げたものとほぼ同様で，$M = 2^{\lfloor \delta n \rfloor}$ および $\tilde{d_k} = d \bmod M$ を計算していることと $F_k(x_k, y)$ の構成のみが異なる．

パラメータ $\ell = 3$ および $m = 2$ として，1024 ビット RSA に対する実験をいくつかの β, δ に対して 1 回ずつおこなった．このとき，格子の次元は 108 となり，一回の計算時間は 10 から 15 時間程度となった．実験結果を図としてまとめたものを図 3 に示す．横軸と縦軸はそれぞれ β および δ をあらわす．図中の＋と×がそれぞれ実験をおこなったパラメータであり，＋が実験成功を，×が失敗を示している．特に，$\beta = 1$ のとき $\delta \geq 0.94$ の範囲において実験が成功している．

$(\ell, m) = (3, 2)$ に対して式 (14) を計算すると $2\beta - \delta - 0.928 < 0$ となり，図中の太線よりも上の領域を示している．これは理論的には，$\beta = 1$ のときには十分な秘密鍵の露出があったとしても攻撃が成功しないことを示している．これに反して，太線よりもかなり下の部分でも実験が成功している

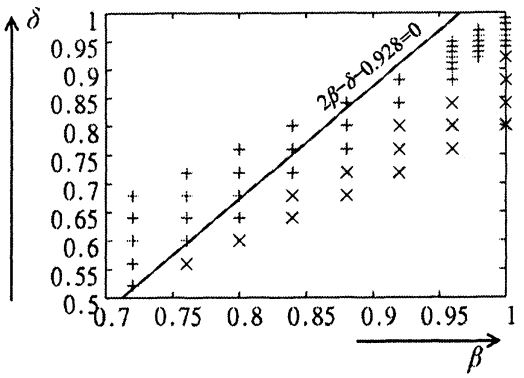

図 3: 1024 ビット RSA に対する部分秘密鍵露出攻撃の実験結果

ことから，構成した格子の部分格子で良いものが存在し，その基底が LLL アルゴリズムにより自動的に選ばれたものと考えられる．これはたとえば Boneh-Durfee の論文 [1] において $\beta < 0.284$ を実現する格子の部分格子を取り出し，それが $\beta < 0.292$ を達成することを証明していることなどから，同様に解析を進めていけば実験値と合うような理論的解析ができると思われるが，現状では未解決である．

References

[1] D. Boneh and G. Durfee, Cryptanalysis of RSA with private Key d Less Than $N^{0.292}$, in *Proceedings of EUROCRYPT 1999*, LNCS, 1592, pp. 389-401, 1999.

[2] D. Boneh, G. Durfee and Y. Frankel, Exposing an RSA private key given a small fraction of its bits *IEEE Transactions on Information Theory*, in *Proceedings of ASIACRYPT* 1998, LNCS, vol. 1514, pp. 25-34, 1998.

[3] J. Blömer and A. May, New partial key exposure attacks on RSA, in *Proceedings of CRTPTO 2003*, LNCS, vol. 2729, pp. 27-43, 2003.

[4] D. Coppersmith, Finding a small root of a univariate modular equation, *Proceedings of EUROCRYPT 1996*, LNCS, vol. 1070, pp. 155-165, 1996.

[5] D. Coppersmith, Finding a small root of a bivariate integer equation; factoring with high bits known, *Proceedings of EUROCRYPT 1996*, LNCS, vol. 1070, pp. 178-189, 1996.

[6] J. S. Coron and A. May, Deterministic polynomial-time equivalence of computing the RSA secret key and factoring, *Journal of Cryptology*, vol. 20,No. 1, pp. 39-50, 2007.

[7] The GNU MP Bignum Library, available online at `http://gmplib.org/`.

[8] GiNaC is Not a CAS, available online at `http://www.ginac.de/`.

[9] N. Howgrave-Graham, Finding small roots of univariate modular equations revisited, *Proceedings of Cryptography and Coding*, LNCS, vol. 1355, pp. 131-142, 1997.

[10] N. Howgrave-Graham and J. P. Seifert, Extending Wiener's attack in the presence of many decrypting exponents, *Proceedings of Secure Networking 1999*, LNCS, vol. 1740, pp. 153-166, 1999.

[11] E. Jochemz and A. May, A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants, in *Proceedings of ASIACRYPT 2006*, LNCS, vol. 4284, pp. 267-282, 2006.

[12] A. K. Lenstra, H. W. Lenstra, Jr. and L. Lovász Factoring polynomials with rational coefficients, *Mathematische Annalen*, vol. 261, pp. 515-534, 1982.

[13] V. Shoup, NTL: A Library for doing Number Theory, available online at `http://www.shoup.net/ntl/index.html`

[14] 野呂正行，横山和弘，グレブナー基底の計算 基礎篇，東京大学出版会，2003.

[15] R. L. Rivest, A. Shamir, and L. Adleman, A method for obtaining digital signatures and public-key cryptsystems, *Communications of the ACM*, vol. 21, No. 2, pp. 120-128, 1978.

[16] S. Sarkar and S. Maitra, Cryptanalysis of RSA with two decryption exponents, in *Information Processing Letter*, Vol. 110, pp. 178-181.

[17] S. Sarkar and S. Maitra, Cryptanalysis of RSA with more than one decryption exponent, in *Information Processing Letter*, Vol. 110, pp. 336-340.